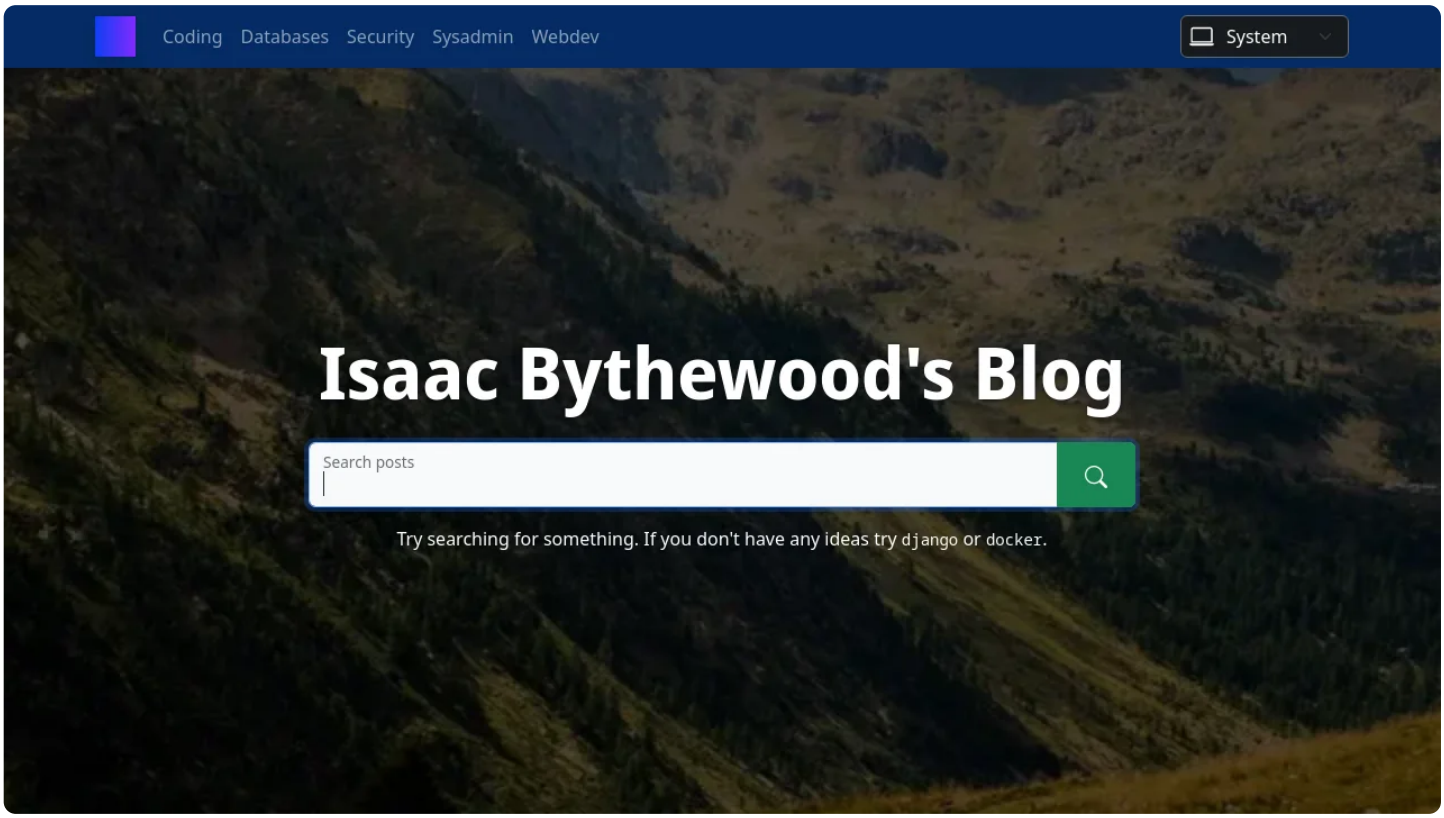
Coding

3 min read

Capturing screenshots with Chromium using Python

Sometimes you need to take screenshots of the web and Chromium provides an easy way to do that.



Author

Isaac Bythewood

August 06, 2022

Chromium for a long time has provided a CLI for capturing web screenshots. I've found myself recently needing a to do a lot of this.

To start my script I import my deps, find Chromium, and setup my base command. I've found that Chromium can be under two different names, `chromium` and `chromium-browser`, depending on your container OS so the path check helps with that.

This example also makes use of Django's `default_storage` functionality to store files in the proper location making this work with a variety of different storage options.

```
import distutils
import os
import subprocess
import uuid

from django.core.files.storage import default_storage


# Get chromium path, it's sometimes chromium and sometimes chromium-brow
chromium = None
if distutils.spawn.find_executable("chromium"):
    chromium = "chromium"
elif distutils.spawn.find_executable("chromium-browser"):
    chromium = "chromium-browser"
else:
    raise Exception("Could not find chromium")


base_command = [
    chromium,
    "--headless",
    "--no-sandbox",
    "--use-gl=swiftshader",
    "--disable-gpu",
    "--disable-software-rasterizer",
    "--disable-dev-shm-usage",
    "--disable-crash-reporter",
    "--disable-extensions",
    "--disable-in-process-stack-traces",
    "--disable-logging",
```

```
    "--window-size=1280x720",
    "--hide-scrollbars",
]
```

Note that I do use Chromium in a Docker container for this so I have a flag that disables Chromium sandboxing since that's the current recommended way of running Chromium inside Docker. You should absolutely remove this flag if you aren't running Chromium in a container.

I then make two helper functions for saving images to storage and running our Chromium command, you can modify this to save to the OS directly if you don't want to use Django's storage system.

```
def save_tempfile_to_storage(tempfilename, filename):
    """
    Saves the given tempfile to django default_storage.
    :param tempfilename: The tempfile we want to save
    :param filename: The storage location to save the file to
    """
    if default_storage.exists(filename):
        default_storage.delete(filename)
    default_storage.save(filename, open(tempfilename, "rb"))
    os.remove(tempfilename)


def run_chromium_command(command):
    """
    Runs the given chromium command and returns the stdout.
    :param command: The command to run
    """
    command = command.split()
    command = base_command + command
    subprocess.run(
        command, check=True, stdout=subprocess.DEVNULL, stderr=subproces
    )
```

Then create our two main functions for generating the actual screenshots, one for generating from a URL and one from generating from HTML directly. You'll also need to modify these slightly if you don't want to use Django's storage system.

```
def generate_screenshot_from_url(url, filename):
    """
    Generates a screenshot of the given url and saves it to the given ou
    file.
    :param url: The url to screenshot
    :param filename: The output file to save the screenshot to
    """
    tempfilename = f"{uuid.uuid4()}.png"
    run_chromium_command(f"--screenshot={tempfilename} {url}")
    save_tempfile_to_storage(tempfilename, filename)
    return default_storage.url(filename)


def generate_screenshot_from_html(html, filename):
    """
    Generates a screenshot of the given html and saves it to the given o
    file.
    :param html: The html to screenshot
    :param filename: The output file to save the screenshot to
    """
    tempfilename = f"{uuid.uuid4()}.html"
    with open(tempfilename, "w") as f:
        f.write(html)
    tempfilename_path = "file://" + os.path.join(os.getcwd(), tempfilena
    run_chromium_command(f"--screenshot={tempfilename} {tempfilename_pat
    save_tempfile_to_storage(tempfilename, filename)
    return default_storage.url(filename)
```

You can now import these two functions anywhere you want to create a screenshot. As a quick example if you wanted to take a screenshot of my blog you'd run:

```
```

```
from chromium import generate_screenshot_from_url

generate_screenshot_from_url("https://blog.bythewood.me/", "screenshots/
```

As a bonus if you wanted to generate a PDF you can add another function to do this very easily since Chromium supports CLI PDF generation.

```
def generate_pdf_from_url(url, filename):
    """
    Generates a pdf of the given url and saves it to the given output fi
    :param url: The url to screenshot
    :param filename: The output file to save the screenshot to
    """
    tempfilename = f"{uuid.uuid4()}.pdf"
    run_chromium_command(f"--print-to-pdf-no-header --print-to-pdf={temp
    save_tempfile_to_storage(tempfilename, filename)
    return default_storage.url(filename)
```

You'd run this the exact same way as the `generate_screenshot_from_url` function.

That's all you need to generate screenshots and PDFs! I've found this to be much more consistent than using the various screenshot and PDF libraries available for Python, you also have a lot of control over Chromium with it's many CLI switches.
